Fachhochschule Dortmund
University of Applied Sciences and Arts

we focus on students

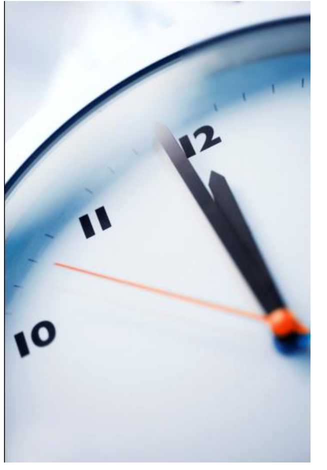
Feedback-aware Requirements Documents for Smart Devices

Erik Kamsties, Fabian Kneer, Markus Voelter, Burkhard Igel, Bernd Kolb
[1]University of Applied Sciences and Arts, Dortmund, Germany
[2]independent/ itemis
[3]itemis AG, Germany

pimes
PROCESS IMPROVEMENT FOR MECHATRONIC AND EMBEDDED SYSTEMS

Overview

- Background
- Motivation
- Goal and Approach
- Representation of requirements
  - Development Time
  - Runtime
- Requirements Monitoring
- Requirements Feedback
- Conclusion



Background

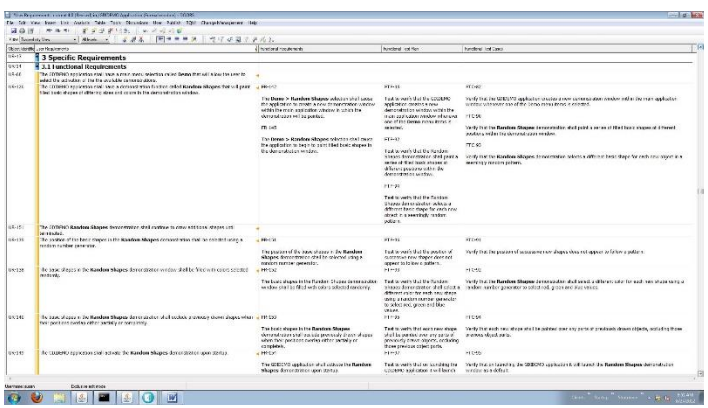
- Smart device are software-intensive systems which
  - operate autonomously
  - interacts with other systems over wireless connections
  - faced with uncertainty in the environment
  - limited resources (CPU, memory)
- Example: eCall adds connectivity to a vehicle

[http://ec.europa.eu/digital-agenda/en/ecall-time-saved-lives-saved]

[Continental]

Motivation

- Runtime representations of requirements allow for
  - reasoning about the requirements at runtime
  - adapting the configuration of a system according to changes in the environment
- Problem: There is no connection between
  - development time requirements (SRS) and
  - runtime, dynamic requirements model inside a system

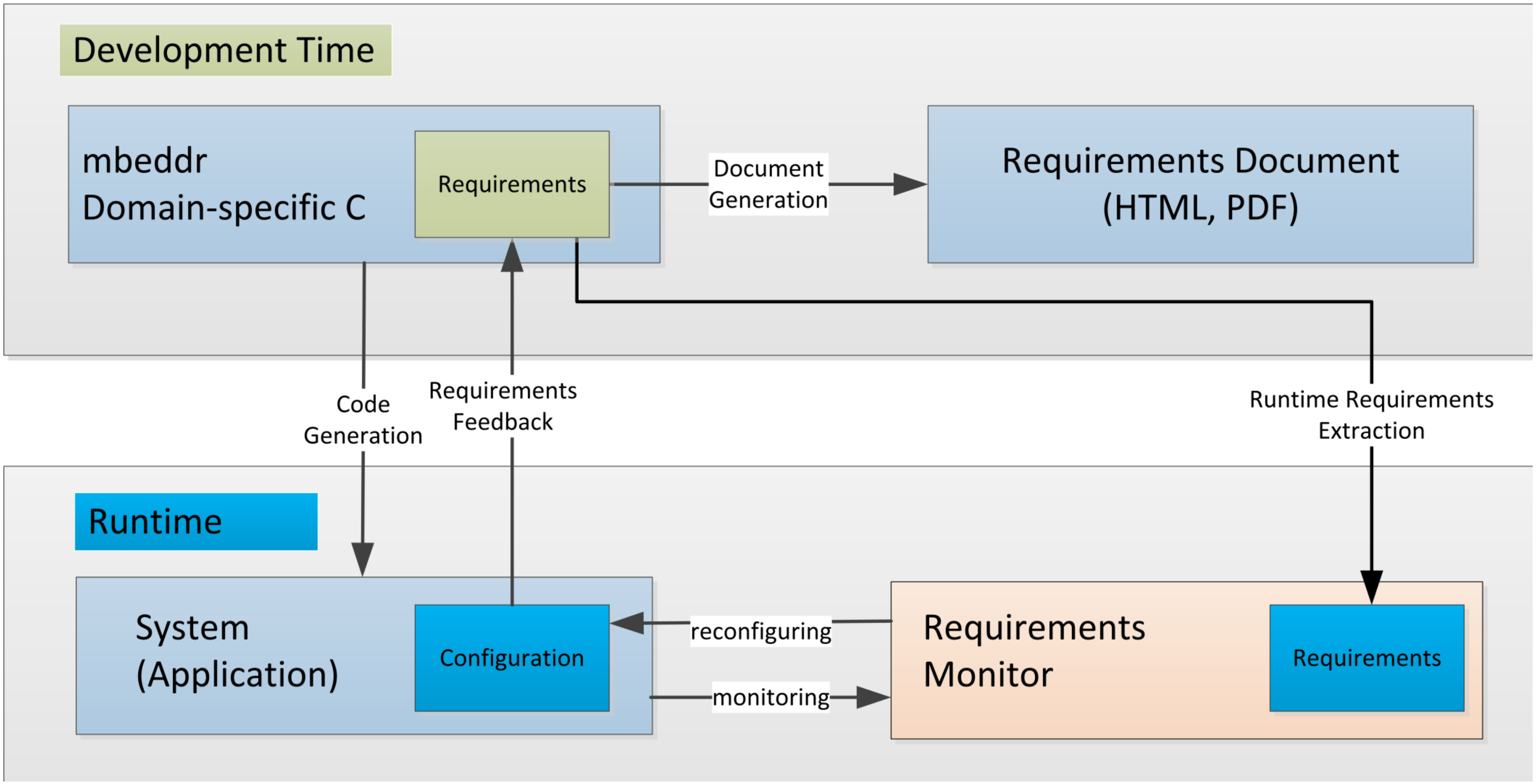


Goal and Approach

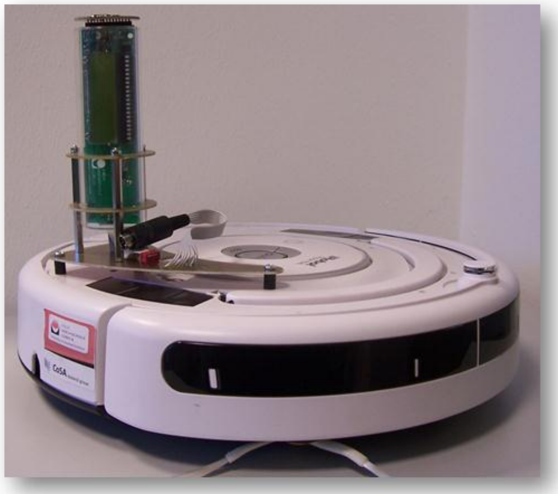
- Bridging the gap between development time and runtime representations of requirements
  - Engineers: better understanding of environment and users
  - Users: Better understanding of the system

- Approach:
  - *Generate* a runtime requirements model out of development time requirements
  - *Monitor* requirements and *compute* reconfigurations
  - *weave* feedback from the runtime system into requirements documents

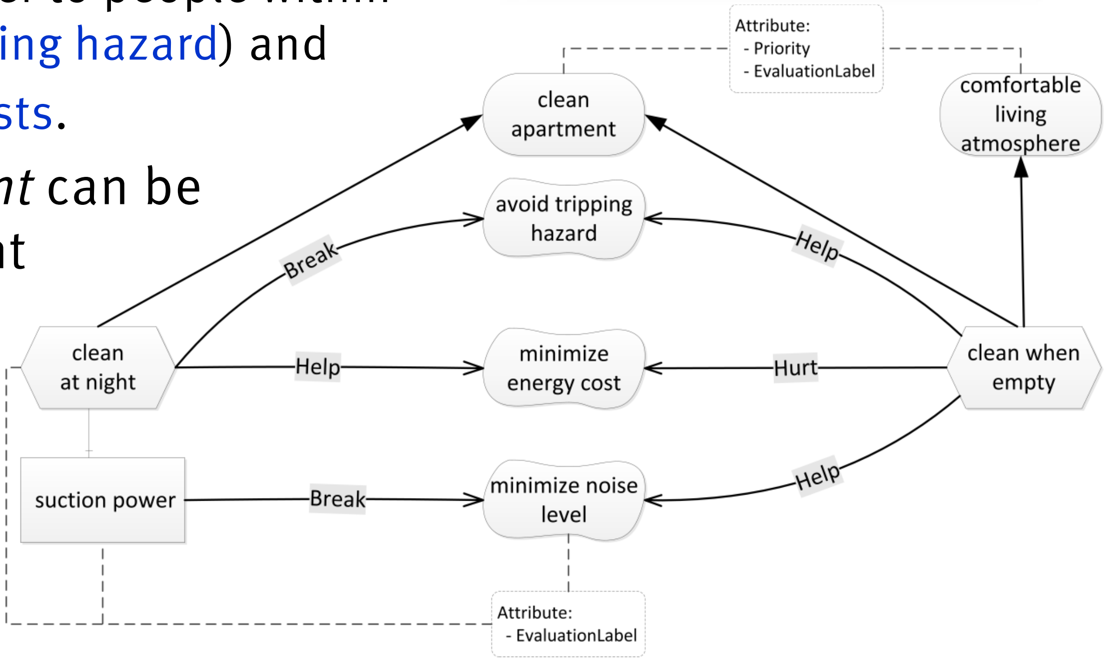
Goal and Approach



Vacuum Cleaner Example [1]

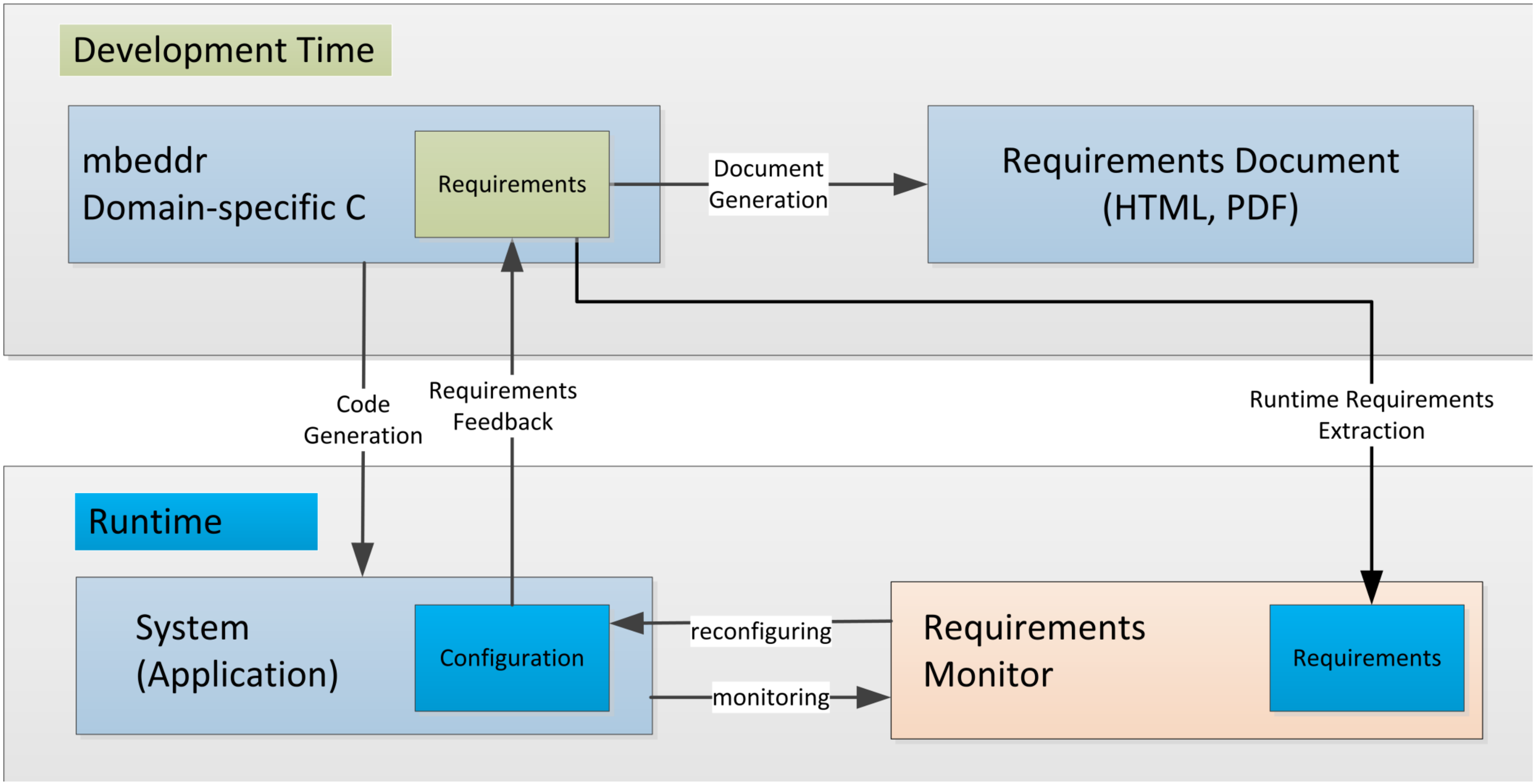


- Vacuum cleaner has two goals
  - to clean apartment and
  - to ensure comfortable living
- two soft goals
  - to avoid causing danger to people within the house (avoid tripping hazard) and
  - to minimize energy costs.
- The goal *clean apartment* can be satisfied by two different realization strategies
  - clean at night or
  - clean when empty

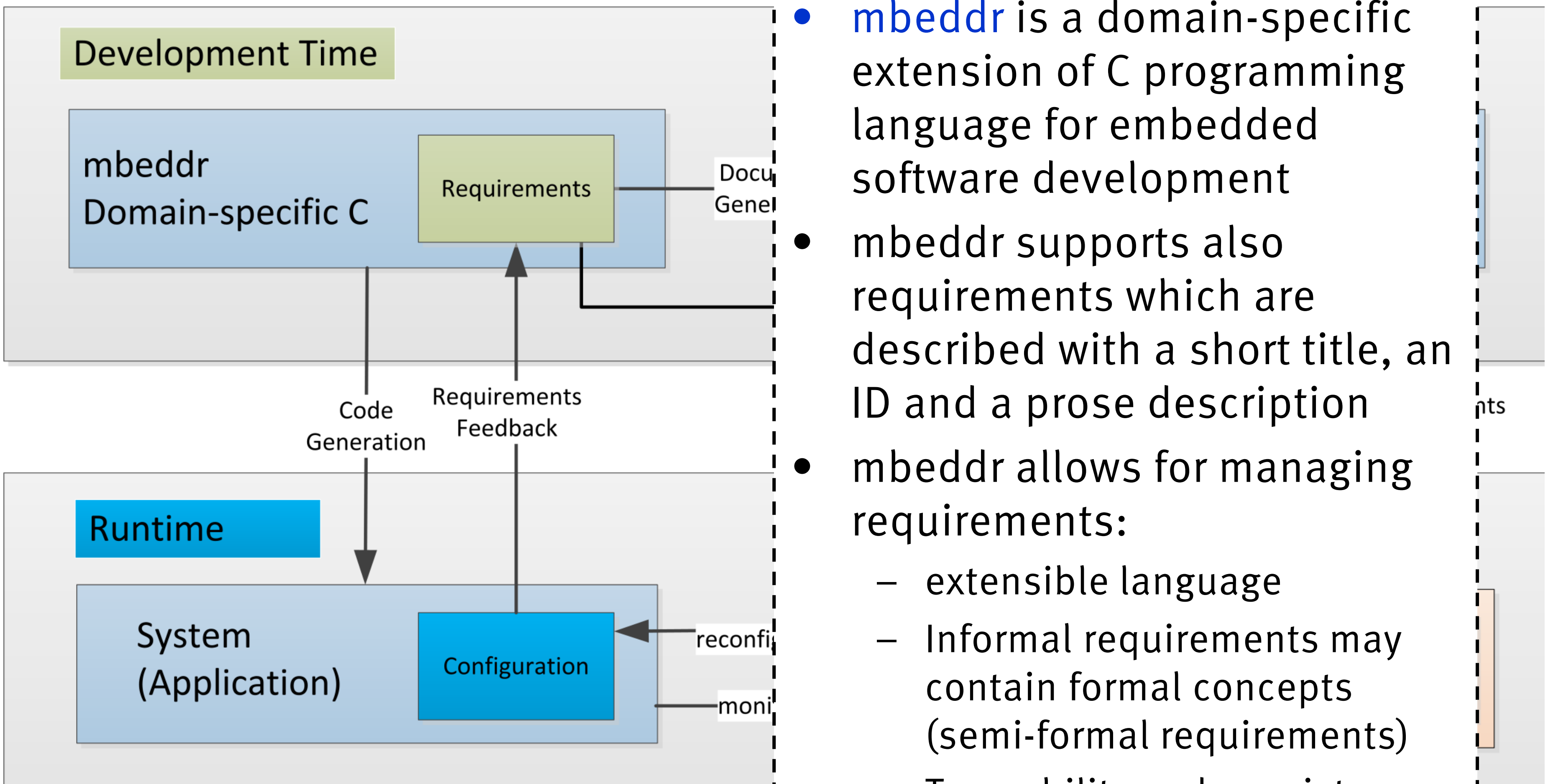


[1] Bencomo, N. and Belaggoun, A., Supporting decision-making for self-adaptive systems, in REFSQ 2013.

Development Time Requirements



Development Time Requirements



- mbeddr is a domain-specific extension of C programming language for embedded software development
- mbeddr supports also requirements which are described with a short title, an ID and a prose description
- mbeddr allows for managing requirements:
  - extensible language
  - Informal requirements may contain formal concepts (semi-formal requirements)
  - Traceability and consistency

Development Time Requirements

1 Clean at night

RE1 /functional: option

[The robot shall clean the apartment at night.]

2 Clean when empty

RE2 /functional: option

[The robot shall clean the apartment when nobody is inside.]

3 Minimize noise level

RE3 /functional: tags

[The robot shall work with a reduced suction power (lower than 50%), so $param(maxSuction: int32 = 50).]

(Application)

Configuration

moni

nts

ecific

ming

itle, an

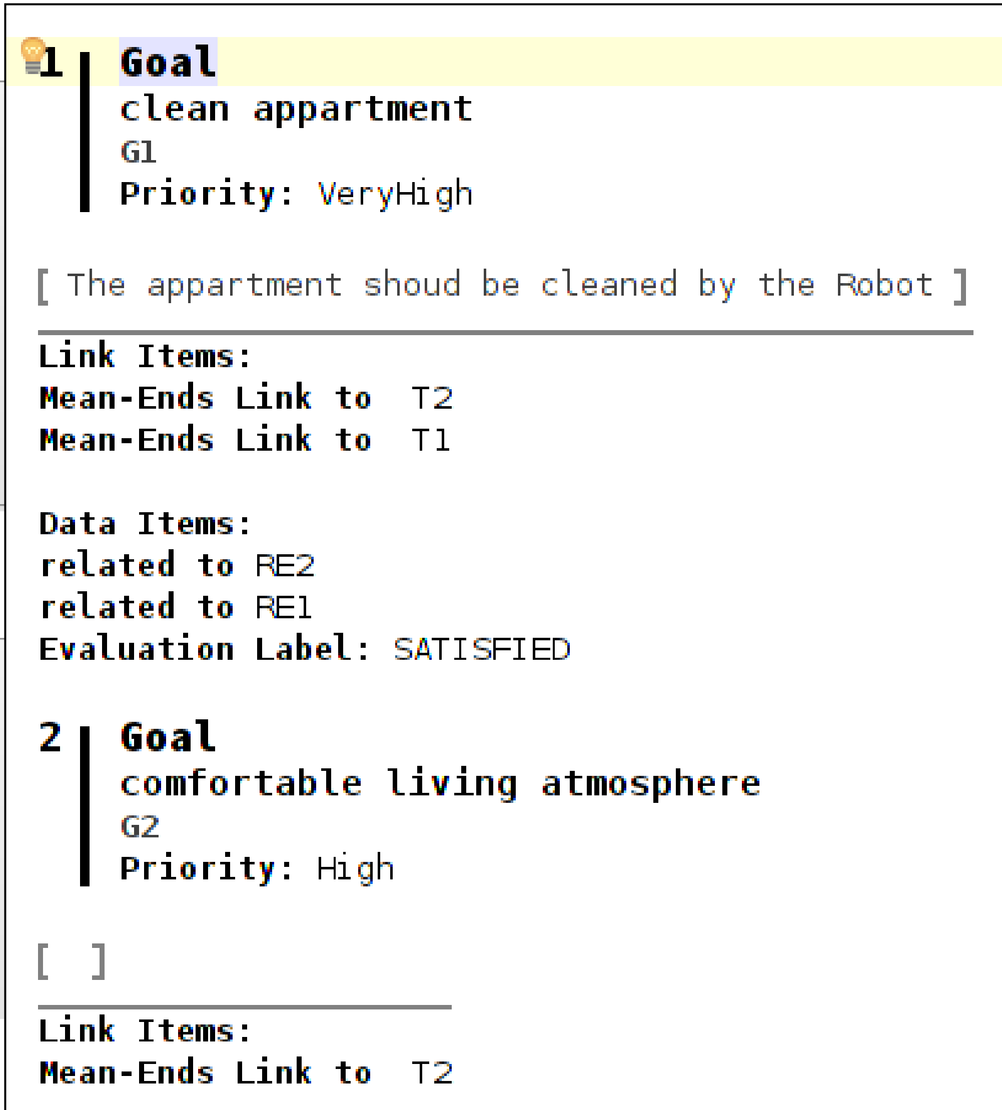
ion

aging

s may contain formal concepts (semi-formal requirements)

– Traceability and consistency

Development Time Requirements



nbeddr is a domain-specific xtension of C programming anguage for embedded oftware development

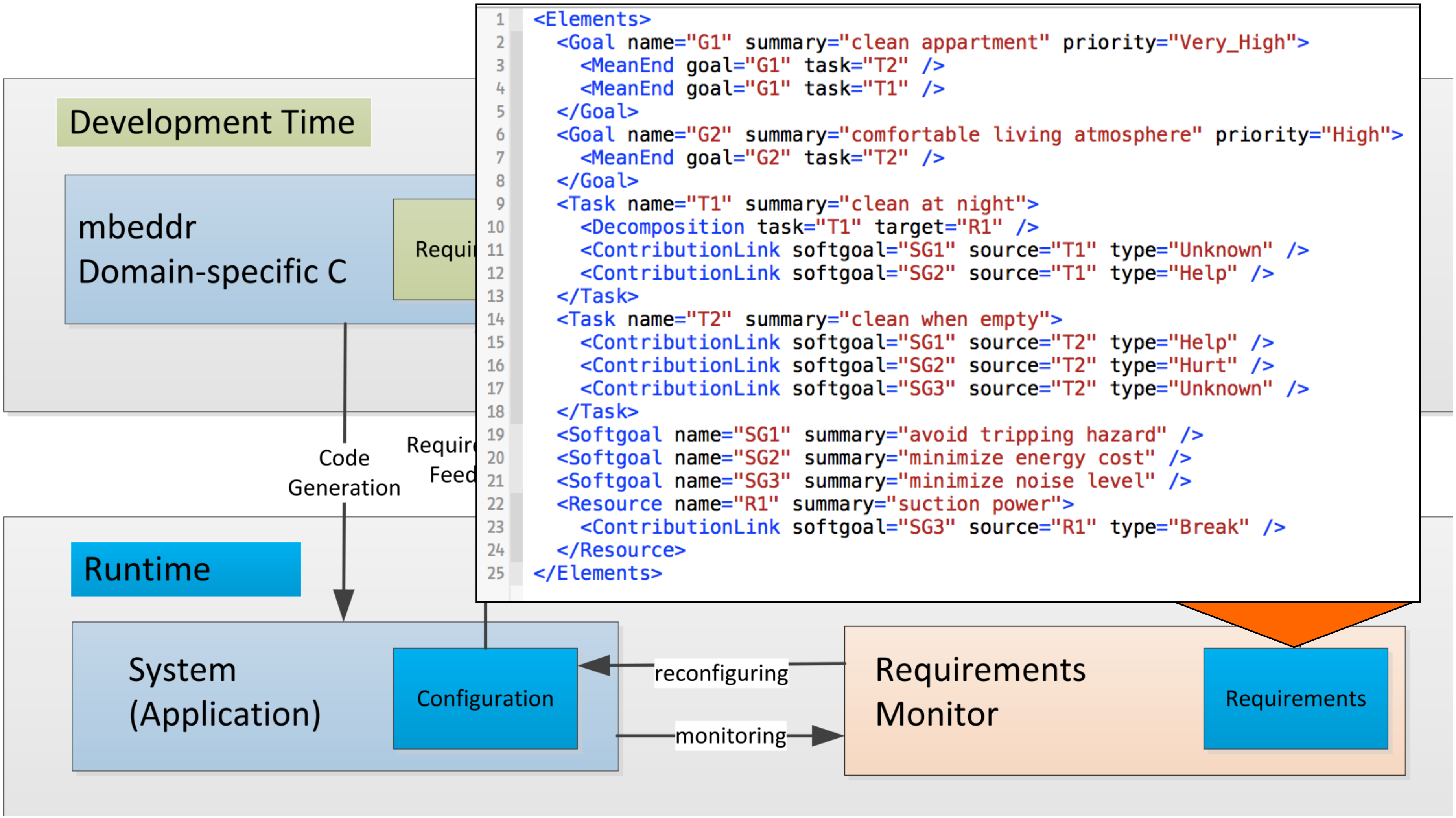
nbeddr supports also equirements which are lescribed with a short title, an D and a prose description

nbeddr allows for managing equirements:

- extensible language
- Informal requirements may contain formal concepts (semi-formal requirements)
- Traceability and consistency

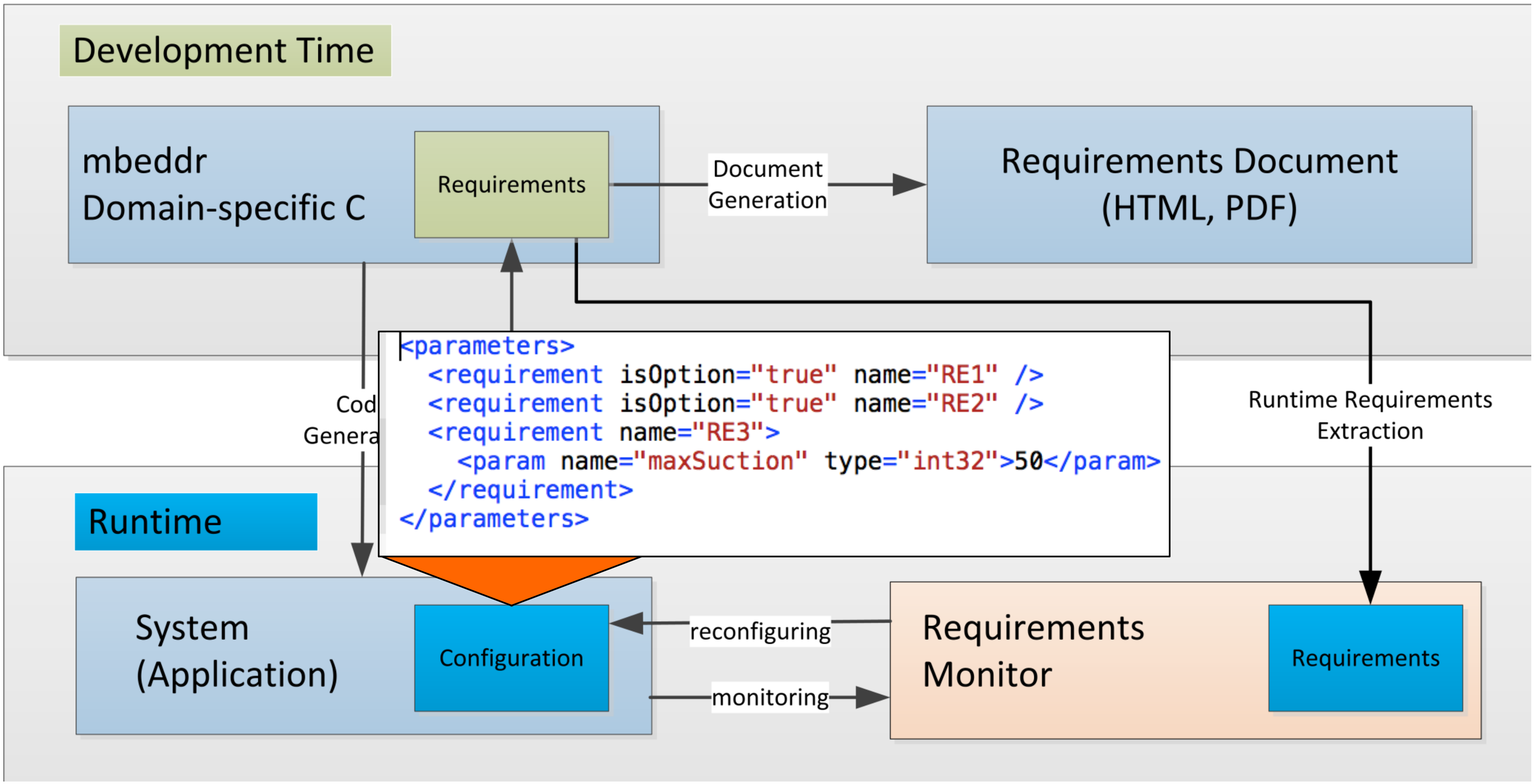
nts

Runtime Requirements




```xml
<Elements>
  <Goal name="G1" summary="clean appartment" priority="Very_High">
    <MeanEnd goal="G1" task="T2" />
    <MeanEnd goal="G1" task="T1" />
  </Goal>
  <Goal name="G2" summary="comfortable living atmosphere" priority="High">
    <MeanEnd goal="G2" task="T2" />
  </Goal>
  <Task name="T1" summary="clean at night">
    <Decomposition task="T1" target="R1" />
    <ContributionLink softgoal="SG1" source="T1" type="Unknown" />
    <ContributionLink softgoal="SG2" source="T1" type="Help" />
  </Task>
  <Task name="T2" summary="clean when empty">
    <ContributionLink softgoal="SG1" source="T2" type="Help" />
    <ContributionLink softgoal="SG2" source="T2" type="Hurt" />
    <ContributionLink softgoal="SG3" source="T2" type="Unknown" />
  </Task>
  <Softgoal name="SG1" summary="avoid tripping hazard" />
  <Softgoal name="SG2" summary="minimize energy cost" />
  <Softgoal name="SG3" summary="minimize noise level" />
  <Resource name="R1" summary="suction power">
    <ContributionLink softgoal="SG3" source="R1" type="Break" />
  </Resource>
</Elements>
```

Runtime Requirements



Requirements Monitoring

- The runtime requirements (i.e., soft goals and contribution links) are connected to assertions in the system

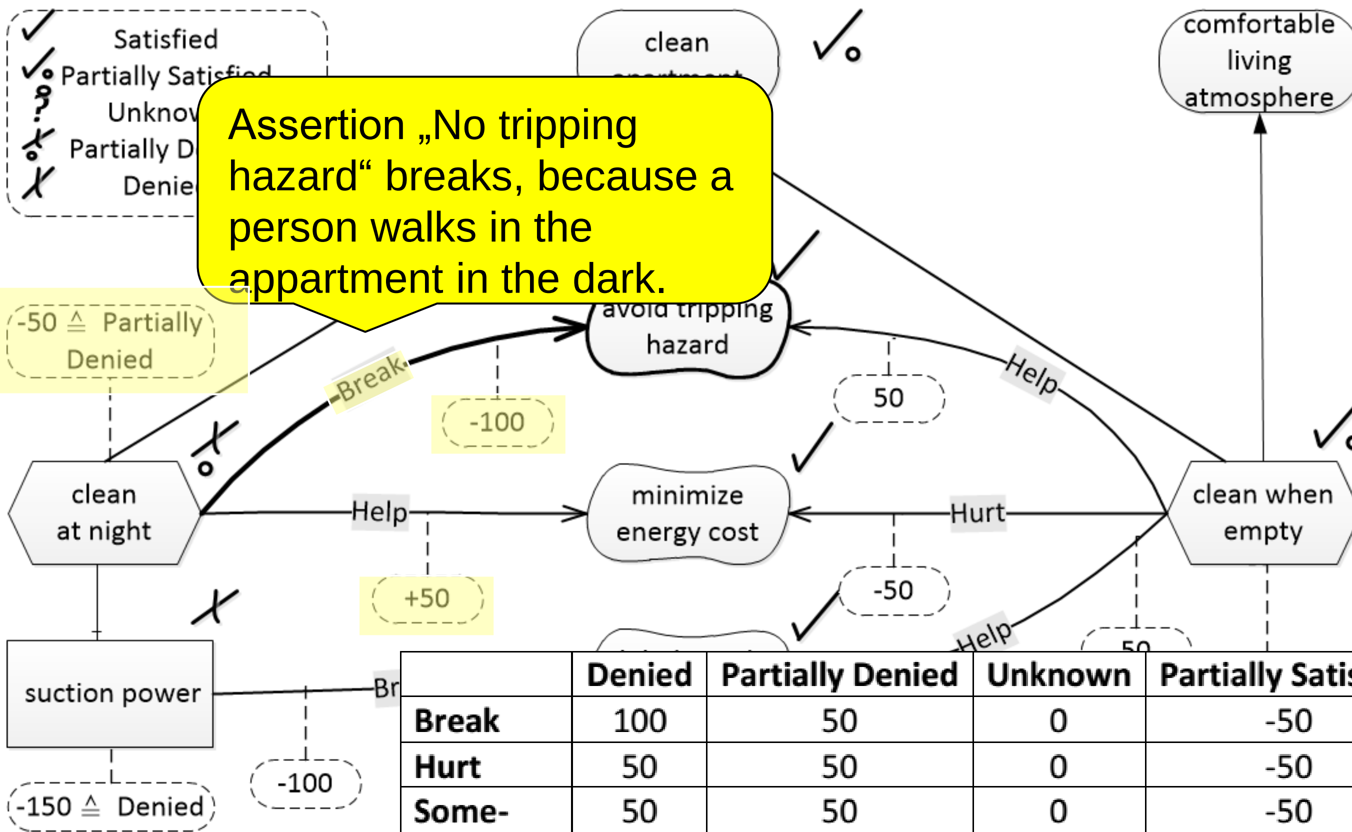
Listing 2. Implementation of a rule with Roolie

```
boolean passes = time > timeMin && time < timeMax;
```

- When an assertion breaks, the runtime requirements are reevaluated and a new configuration is computed (on particular conditions, reconfiguration can be delayed)

- For this purpose we extended the i* model slightly by the concept of a priority (of a goal) and we extended the i* model evaluation process by Grau et al.

Requirements Monitoring



	Denied	Partially Denied	Unknown	Partially Satisfied	Satisfied
Break	100	50	0	-50	-100
Hurt	50	50	0	-50	-50
Some-	50	50	0	-50	-50
Unknown	0	0	0	0	0
Some+	-50	-50	0	50	50
Help	-50	-50	0	50	50
Make	-100	-50	0	50	100

Requirements Feedback

Requirements vacuum_c

doc config: test
class: runtime-requir

Abstract: [ ]


```xml
<parameters-feedback systemID="system_42">
  <requirement isOption="true" name="RE1" />
  <requirement isOption="false" name="RE2" />
  <requirement name="RE3">
    <param name="maxSuction" type="int32">80</param>
  </requirement>
</parameters-feedback>
```


1 Clean at night

RE1 /functional: option=false

[The robot shall clean the apartment at night.]

2 Clean when empty

RE2 /functional: option=true

[The robot shall clean the apartment when nobody is inside.]

3 Minimize noise level

RE3 /functional: tags

[The robot shall work with a reduced suction power (lower than 50%), so **$param**(maxSuction = 50).]

Conclusion

- The goal of our work is to gain insights into how requirements evolve over time and how the system is actually used.
- We proposed an approach to establish the missing link between development time and runtime representations of requirements in the context of embedded systems.
- Smart/embedded systems are particularly interesting, as a human operator is often not available to give a confirmation to a system's decision. Thus the system must decide autonomously.
- I have a demonstrator with me to give a practical demo in a break…